**Agenda**

* Jeff continued presenting MPI requirements

**Jeff’s slides – libfabric-old-mpi-presentatoins-2014-01-14.pptx**

Covered prediction of MPI needs for verbs moving to exascale. See slides for full details. Select notes captured below.

* May need to take advantage of MPI’s addressing model – group + logical address
Basically, cannot rely on storing O(n) data storage.
* Must assume that there will be faults. This was pushed out of MPI-3 requirements.
* Someone must handle retries. This could be pushed into the network or the network can provide the necessary mechanisms for MPI to implement this.
* MPI dislikes needing to intercept calls in order to do memory deregistration.
* Desire to export the network topology to MPI.

Question was whether most MPI apps used swap space. The general response was no, but this depends on the scale of the cluster. For very small clusters (e.g. 32-128 nodes), swap space may be turned on.

Question was how MPI loaded its data sets. Jeff answered that they were usually read from files. Each process may read in their own data or one may read in the data and use MPI calls to distribute the data. File access is usually NOT done using memory mapping.

**Next presentation – 2014-01-21-mpi-community-feedback.pptx**

MPI Requirements of the Network Layer. Select notes captured below.

* No concept of connection – send specifies remote address.
* Ordering rules are based on groups of message wrt each other.
* Transfers are message based. Streams not currently defined.
* MPI wants asynchronous progress. Ideally, the HW provides this, but that is not required. MPI wants progress done outside of calling some poll routine.
* MPI typically copies small messages into bounce buffers. For larger buffers, MPI will register the buffer and store that in a cache. If registration continues to exist, MPI would like an asynchronous registration call.
* For structured applications, many of those allocate their memory during setup. But more complex apps do not have this characteristic.
* Question was whether the system could intercept all malloc and map calls and automatically register those buffers. Response is that this would quickly run out of resources or result in poor performance trying to cache all these registrations. There would need to better coordination between the NIC and IOMMU.

Request to create an MPI libfabric listserv.